

What is claimed is:

- 1 1. A method comprising:
2 configuring processors with multiple threads of
3 execution to execute a critical section of code, the
4 processors operable to execute the critical section in turns;
5 and
6 controlling the threads of execution of the processors to
7 avoid occurrence of idle time between execution of the
8 critical section by the processors.

- 1 2. The method of claim 1 wherein controlling comprises:
2 enabling the threads of execution on each of the processors to
3 execute in order via inter-thread signaling.

- 1 3. The method of claim 2 wherein enabling comprises:
2 enabling each thread, when such thread is executing, to
3 execute a first instruction to cause an inter-thread signal to
4 a next thread to be generated; and
5 enabling the thread to execute a second instruction, which
6 causes the thread to wait for an inter-thread signal from a
7 previous thread, only after a write latency required by the
8 first instruction.

1 4. The method of claim 3 wherein the write latency comprises
2 at least three instruction cycles.

1 5. The method of claim 3 wherein the processors each comprise
2 a register through which an inter-thread signal to the next
3 thread is given.

1 6. The method of claim 5 wherein the processors each comprise
2 registers through which inter-thread, inter-processor
3 signaling can occur.

1 7. The method of claim 6 wherein the processors use external
2 registers to enable inter-thread, inter-processor signaling to
3 occur.

1 8. The method of claim 2 wherein controlling further
2 comprises:
3 enabling each thread of execution on each of the processors to
4 relinquish control of a program comprising the critical
5 section as soon as the critical section has been executed.

1 9. The method of claim 1 wherein controlling comprises:
2 enabling each thread of execution on each of the processors
3 to relinquish control of the program comprising the critical
4 section as soon as the critical section has been executed.

1 10. The method of claim 1 wherein the processors are operable
2 to execute at least two of the critical sections of code.

1 11. The method of claim 10 wherein the processors comprise a
2 functional pipeline in a network processor.

1 12. The method of claim 11 wherein one of the critical
2 sections of code comprises a metering microblock and the
3 another of the critical sections of code comprises a
4 congestion avoidance microblock.

1 13. The method of claim 1 wherein the processors comprise a
2 functional pipeline, and one or more of the critical sections
3 of code are executed in the functional pipeline.

1 14. The method of 13 wherein one of the one or more critical
2 sections of code comprises an Asynchronous Transfer Mode (ATM)
3 receive processing microblock.

1 15. The method of claim 13 wherein one of the one or more
2 critical sections comprises an ATM traffic management
3 processing microblock.

1 16. A article comprising:
2 a storage medium having stored thereon instructions that
3 when executed by a machine result in the following:
4 configuring processors with multiple threads of
5 execution to execute a critical section of code, the
6 processors operable to execute the critical section in turns;
7 and
8 controlling the threads of execution of the processors to
9 avoid occurrence of idle time between execution of the
10 critical section by the processors.

1 17. The article of claim 16 wherein controlling comprises:
2 enabling the threads of execution on each of the processors to
3 execute in order via inter-thread signaling.

1 18. The article of claim 17 wherein enabling comprises:
 2 enabling each thread, when such thread is executing, to
 3 execute a first instruction to cause an inter-thread signal to
 4 a next thread to be generated; and
 5 enabling the thread to execute a second instruction, which
 6 causes the thread to wait for an inter-thread signal from a
 7 previous thread, only after a write latency required by the
 8 first instruction.

1 19. The article of claim 16 wherein controlling comprises:
 2 enabling each thread of execution on each of the processors
 3 to relinquish control of the program comprising the critical
 4 section as soon as the critical section has been executed.

1 20. A network processor comprising:
 2 a processor; and
 3 multi-threaded processors, having threads of execution,
 4 configurable by the processor to execute a critical section of
 5 code and operable to execute the critical section in turns;
 6 and
 7 wherein the threads of execution of the multi-threaded
 8 processors are controllable to avoid occurrence of idle time
 9 between execution of the critical section by the multi-
 10 threaded processors.

1 21. The network processor of claim 20 wherein each thread,
2 when such thread is executing, is controllable to execute a
3 first instruction to cause an inter-thread signal to a next
4 thread to be generated, and the thread is further controllable
5 to execute a second instruction, which causes the thread to
6 wait for an inter-thread signal from a previous thread, only
7 after a write latency required by the first instruction.

1 22. A network processor of claim 20 wherein each thread of
2 execution on each of the multi-threaded processors is
3 controllable to relinquish control of the program comprising
4 the critical section as soon as the critical section has been
5 executed.

1 23. A system comprising:
2 a memory system to store a critical section of code;
3 processors, coupled to the memory system, having multiple
4 threads of execution to execute the critical section of code,
5 the processors operable to execute the critical section in
6 turns; and

7 wherein the threads of execution of the processors are
 8 controllable to avoid occurrence of idle time between
 9 execution of the critical section by the processors.

1 24. The system of claim 23 wherein each thread, when such
 2 thread is executing, is controllable to execute a first
 3 instruction to cause an inter-thread signal to a next thread
 4 to be generated, and the thread is further controllable to
 5 execute a second instruction, which causes the thread to wait
 6 for an inter-thread signal from a previous thread, only after
 7 a write latency required by the first instruction.

1 25. The system of claim 23 wherein each thread of execution
 2 on each of the multi-threaded processors is controllable to
 3 relinquish control of the program comprising the critical
 4 section as soon as the critical section has been executed.